**135. Candy: -**

Hard Accepted: 381.5K Submissions: 900.2K Acceptance Rate: 42.4%

There are n children standing in a line. Each child is assigned a rating value given in the integer array ratings.

You are giving candies to these children subjected to the following requirements:

* Each child must have at least one candy.
* Children with a higher rating get more candies than their neighbors.

Return *the minimum number of candies you need to have to distribute the candies to the children*.

**Example 1:**

**Input:** ratings = [1,0,2]

**Output:** 5

**Explanation:** You can allocate to the first, second and third child with 2, 1, 2 candies respectively.

**Example 2:**

**Input:** ratings = [1,2,2]

**Output:** 4

**Explanation:** You can allocate to the first, second and third child with 1, 2, 1 candies respectively.

The third child gets 1 candy because it satisfies the above two conditions.

**Constraints:**

* n == ratings.length
* 1 <= n <= 2 \* 104
* 0 <= ratings[i] <= 2 \* 104

**Code: -**

class Solution {

public:

    int candy(vector<int>& rating) {

        int n = rating.size();

        vector<int> temp(n,1);

        // for operating on increasing order but in reverse direction

        for(int i=n-1; i>0; --i){

          if(rating[i-1] > rating[i] and temp[i-1] <= temp[i])

            temp[i-1] = temp[i] + 1;

        }

        // for operating on increasing order

        for(int i=0; i<n-1; ++i){

          if(rating[i] < rating[i+1] and temp[i] >= temp[i+1])

            temp[i+1] = temp[i] + 1;

        }

        return accumulate(temp.begin(), temp.end(), 0);

    }

};

**T.C: - O(N)**

**S.C: - O(N)**